Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3

Технології розроблення програмного забезпечення

Тема: Основи проектування розгортання

Виконав
студент групи ІА-34:
Жительний В.В.

Перевірив:
Мягкий М.Ю.

Київ 2025

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Тема:

21. **Online radio station** (iterator, adapter, factory method, facade, visitor, client-server)

Додаток повинен служити сервером для радіостанції з можливістю мовлення на радіостанцію (64, 92, 128, 196, 224 kb/s) в потоковому режимі; вести облік підключених користувачів і статистику відвідувань і прослуховувань; налаштувати папки з піснями і можливість вести списки програвання або playlists (не відтворювати всі пісні).

Хід роботи

Створення діаграми розгортання:

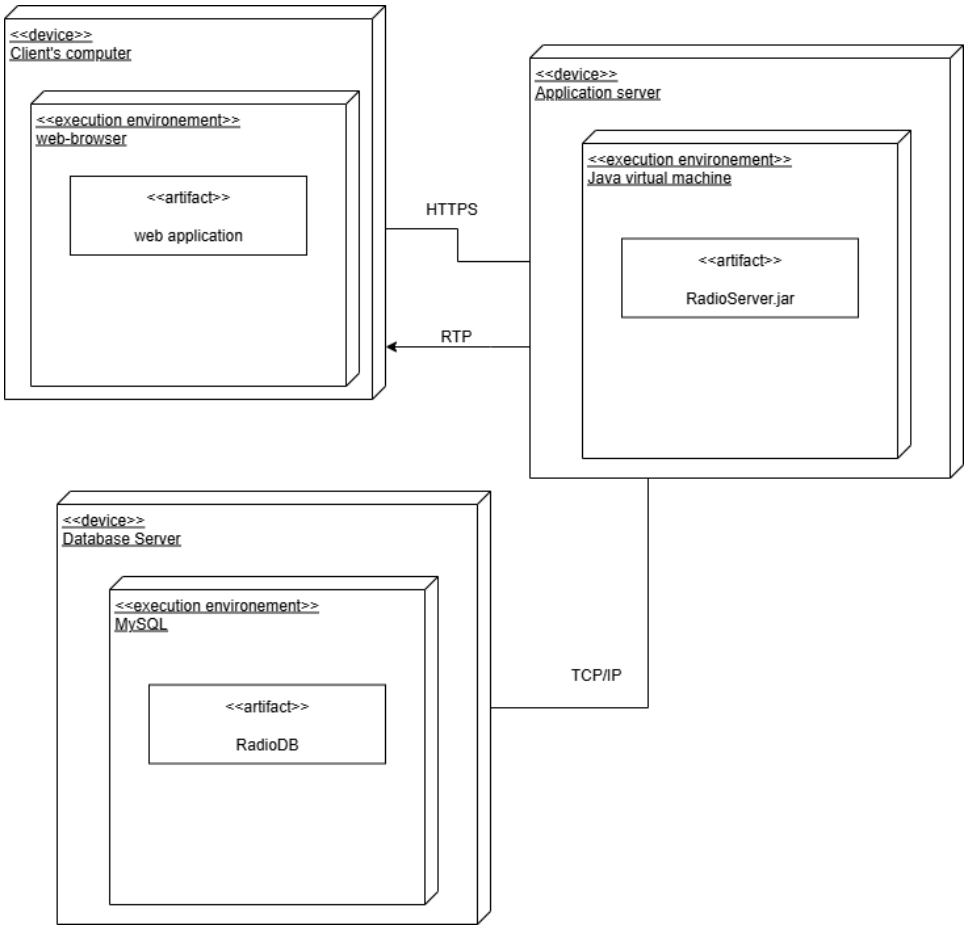

Рис. 1 Діаграма розгортання

Ця діаграма розгортання описує трирівневу архітектуру для онлайн радіо. Вона включає Client's computer тобто клієнтський пристрій, де у веб браузері виконується веб застосунок. Клієнт взаємодіє з сервером додатків за допомогою HTTPS для контролю та отримує потокове медіа через RTP. На сервері додатків працює RadioServer.jar. Сервер додатків підключається через TCP/IP до серверу бази даних, де розміщено базу даних RadioDB під керуванням MySQL.

Створення діаграми компонентів:

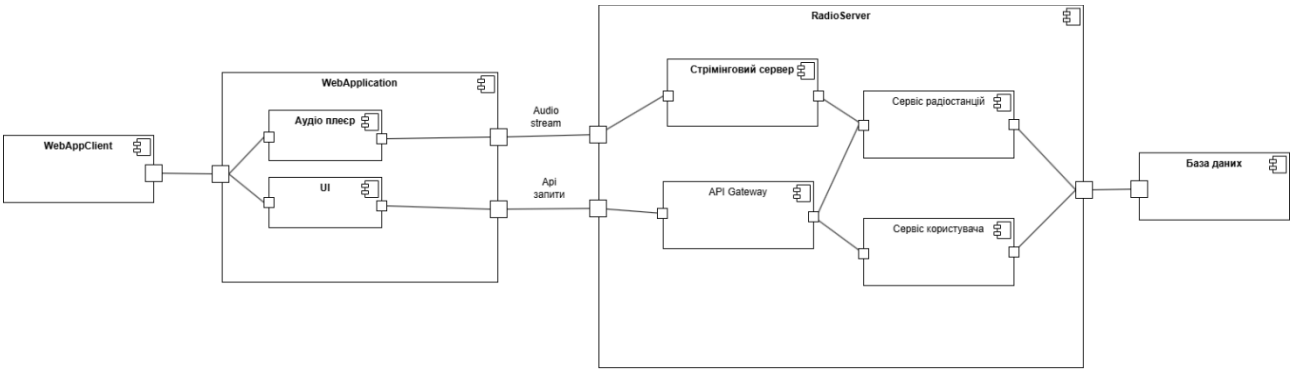

Рис. 2 Діаграма компонентів

Ця діаграма компонентів описує архітектуру системи онлайн-радіо, що складається з компонентів WebAppClient, WebApplication, RadioServer та База даних. Компонент WebApplication, що містить UI та Аудіо плеєр, виступає як посередник, вимагаючи інтерфейси для отримання аудіопотоку та надсилання запитів. Ці інтерфейси надаються компонентом RadioServer, який інкапсулює всю серверну логіку, включаючи API Gateway для обробки запитів, Стрімінговий сервер для передачі медіа, а також Сервіс радіостанцій та Сервіс користувача для реалізації бізнес-логіки. У свою чергу, внутрішні компоненти RadioServer потребують доступу до даних, який забезпечує компонент База даних.

Створення діаграми послідовностей:

Рис. 3 Діаграма послідовностей на перший сценарій

На рисунку 3 зображено діаграму послідовностей для першого сценарію використання з другої лабораторної роботи

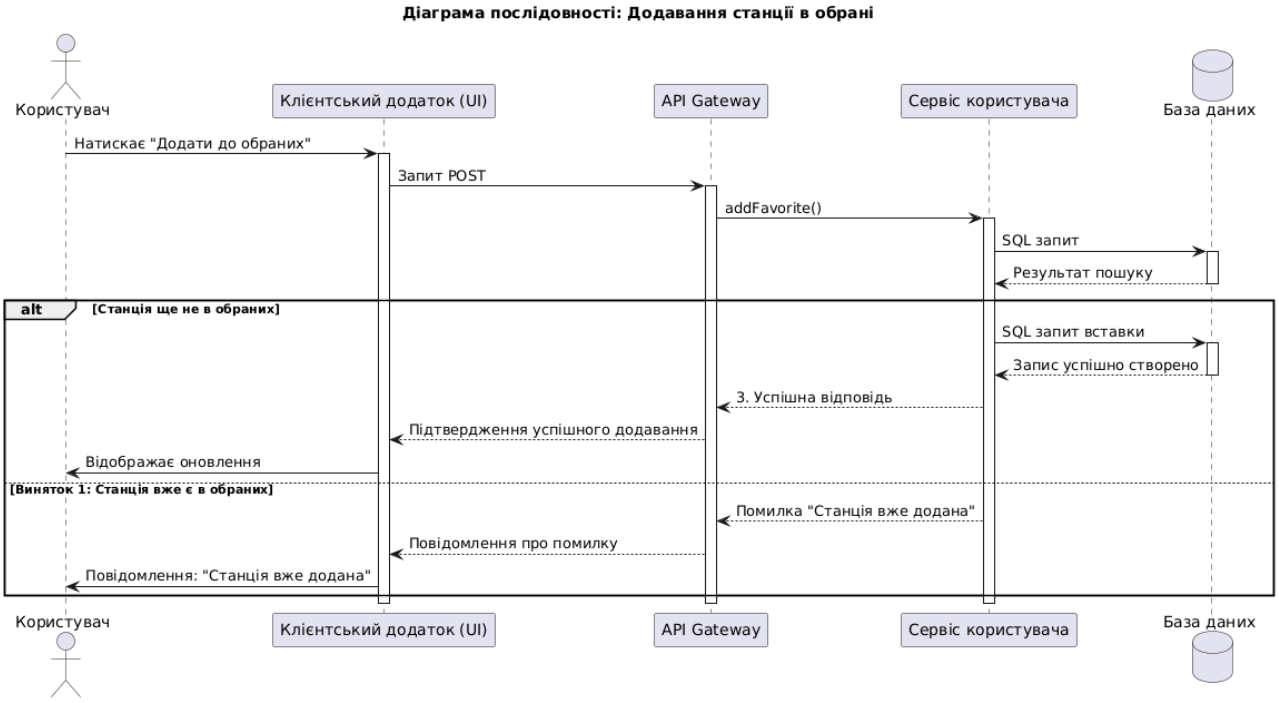

Рис. 3 Діаграма послідовностей на другий сценарій

На рисунку 4 зображено діаграму послідовностей для другого сценарію використання з другої лабораторної роботи

Програмна частина:

Основні класи:

```
public class User implements Element {
    private int id;
    private String username;
    private String password;

    // constructor, getters, setters
}

public class Admin extends User {
    // різні додаткові методи керування станціями
    // constructor, getters, setters
}

public class Station implements Element {
    private int id;
    private String name;
    private List<Playlist> playlists;

    // getters, setters, constructor
}
public class Favorite {
    private int id;
    private int userId;
    private int stationId;

    // constructor, getters, setters
}
```

```
public class Recording {
    private int id;
    private int stationId;
    private int userId;
    private String filePath;
    private LocalDateTime timestamp;
    // getters, setters, constructor
}
```

Фабрики:

```
public class TrackFactory {
    // створює об’єкти Track
    public Track createTrack() {
        return new Track();
    }
}

public class PlaylistFactory {
    // створює об’єкти Playlist
    public Playlist createPlaylist() {
        return new Playlist();
    }
}
```

Visitors:

```
public interface Visitor {
    void visit(User client);
    void visit(Station station);
}
public interface Element {
```

```
    void accept(Visitor visitor);
}

public class Statistics implements Visitor {
    private int totalConnections;
    private List<User> activeClients;
    private Map<User, LocalDate> listeningHistory;

    // constructor, getters, setters
}
```

Реалізація шаблону Repository:

```
public interface UserRepository {
    void addUser(User user);
    User getUserById (int userId);
    void updateUser(User user);
    void deleteUser(int userId);
}

public interface StationRepository {
    void addStation(Station station);
    Station getStationById(int stationId);
    void updateStation(Station station);
    void deleteStation(int stationId);
}

public interface FavoriteRepository {
    void addFavorite(Favorite favorite);
    Favorite getFavoriteById(int favoriteId);
    void updateFavorite(Favorite favorite);
```

```
    void deleteFavorite(int favoriteId);
    List<Favorite> getFavoritesByUserId(int userId);
}

public interface RecordingRepository {
    void addRecording(Recording recording);
    Recording getRecordingById(int recordingId);
    void updateRecording(Recording recording);
    void deleteRecording(int recordingId);
    List<Recording> getRecordingsByUserId(int userId);
}
```

Реалізація шару доступу до даних (На прикладі User):

```
    public class JdbcUserRepository implements UserRepository {
        private final Connection connection;
        public JdbcUserRepository(Connection connection) {
            this.connection = connection;
        }

        @Override
        public void addUser(User user) {
            String sql = "INSERT INTO users (username, password) VALUES (?, ?)";
            //Далі треба буде виконати SQL insert та зберегти згенерований Id у user
        }

        @Override
        public User getUserById(int userId) {
            String sql = "SELECT * FROM users WHERE id = ?";
            // Далі треба буде виконати SQL select і повернути об'єкт User або null
        }

        @Override
        public void updateUser(User user) {
            String sql = "UPDATE users SET username = ?, password = ? WHERE id = ?";
            // Далі треба буде виконати SQL update для даного користувача
        }
```

```
    @Override
    public void deleteUser(int userId) {
        String sql = "DELETE FROM users WHERE id = ?";
        // Далі треба буде виконати SQL delete для даного користувача
    }
}
```

Бізнес логіка:

```
public class UserService {
    private final UserRepository userRepository;
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }
    public void registerNewUser(String username, String password) throws Exception
{
        User existingUser = userRepository.getUserByUsername(username);
        if (existingUser != null) {
            throw new Exception("Користувач з таким іменем вже існує");
        }
        String hashedPassword = hashPassword(password);

        User newUser = new User(username, hashedPassword);
        userRepository.addUser(newUser);
    }
    private String hashPassword(String password) {
        // Тут логіка хешування;
    }
}
```

Відповіді на питання:

1. Що собою становить діаграма розгортання?

Діаграма розгортання – це така діаграма, яка моделює фізичну архітектуру системи. Вона показує як програмні компоненти, які називають артефактами, розміщені на апаратному забезпечені, тобто вузлах, і як ці апаратні частини з’єднані між собою

2. Які бувають види вузлів на діаграмі розгортання?

Всього є два типи вузлів. Пристрій – те, що містить програмне забезпечення, та середовище виконання – програмне забезпечення, яке само може містити інше програмне забезпечення

3. Які бувають зв’язки на діаграмі розгортання?

Є асоція, її використовують для позначення протоколів або технологій, що

використовується для забезпечення взаємодії вузлів.

4. Які елементи присутні на діаграмі компонентів?

На діаграмі компонентів присутні наступні компоненти:

Компонент - Головний елемент, що представляє модуль системи

Інтерфейс - описує послуги

Порт – маленький квадратик, який є точкою для взаємодії із зовнішнім світом

Зв’зки – показують як компоненти взаємодіють між собою

5. Що становлять собою зв’язки на діаграмі компонентів?

Зв'язки показують, як компоненти взаємодіють між собою :

З'єднувач збірки – він показує, що required interface одного компонента задовольняється provided interface іншого. Це основний спосіб демонстрації взаємодії.

Делегування – стрілка, що з'єднує зовнішній порт компонента з його внутрішньою частиною.

6. Які бувають види діаграм взаємодії?

Є два типи таких діаграм:

Діаграма послідовностей та кооперативна діаграма

7. Для чого призначена діаграма послідовностей?

Вона призначена для візуалізації взаємодії об'єктів у хронологічному порядку

8. Які ключові елементи можуть бути на діаграмі послідовностей?

Лінія життя – вона являє собою фрагмент життєвого циклу об'єкта в процесі взаємодії

Повідомлення – стрілка між лініями життя, що показує виклик операції або передачу інформації.

Фрагмент взаємодії – елемент, який дозволяє моделювати складну логіку, наприклад як цикли, умови чи паралельне виконання

9. Як діаграми послідовностей пов’язані з діаграмами варіантів використання?

Діаграма використання показує, що система робить, а діаграма послідовностей деталізує це, показуючи як вона це робить крок за кроком.

10. Як діаграми послідовностей пов’язані з діаграмами класів?

Діаграма класів моделює статичну поведінку, а діаграма послідовностей динамічну

Висновки: У ході роботи я ознайомився та реалізував діаграмами розгортання, компонентів та послідовностей, дізнався як їх реалізовувати та в який випадках їх використовувати.